Contents lists available at SCCE

Journal of Soft Computing in Civil Engineering

Journal homepage: www.jsoftcivil.com

A Modified Genetic Algorithm in C++ for Optimization of Steel Truss Structures

P. Kumar[1*], **S. Pandey**[1], **P.R. Maiti**[2]

1. IDD Student, Department of Civil Engineering, Indian Institute of Technology (IIT) BHU, Varanasi, India
2. Associate Professor, Department of Civil Engineering, Indian Institute of Technology (IIT) BHU, Varanasi, India

Corresponding author: *pawan.kumar.civ15@iitbhu.ac.in*

https://doi.org/10.22115/SCCE.2021.242552.1249

ARTICLE INFO

Article history:
Received: 17 August 2020
Revised: 27 March 2021
Accepted: 19 April 2021

Keywords:
Genetic algorithm;
Optimization operators;
Discreet design variables;
Genotypes.

ABSTRACT

A common structural design optimization problem is weight minimization which is done by choosing a set of variables that represent the structural or the architectural configuration of the system satisfying few design specific criterion. In general, genetic algorithms (GAs) are ideal to be used for unconstrained optimization, so it is required to transform the constrained problem into an unconstrained one. A violation of normalized constraints-based formulation method has been used in the present work for this purpose. A modified algorithm has been developed in C++ using concept of genotypes for optimization using discreet design variable. A detailed analysis of optimization of a simple steel truss with discrete design variables using different variations of genetic algorithm is presented here. Also, an attempt has been made to study the sensitivity of the algorithm with respect to the optimization operators i.e., initial population size, rate of mutation.

1. Introduction

The purpose of the optimization in steel structural design is to determine the optimal cross-sectional area and minimum material consumption for the members considering all other constraints. In practice, it is often desirable that the design variables (such as cross-sectional areas of the member and shape of rolled section) should be chosen from commercially available shape and sizes by the manufacturer. In many cases, the use of an optimization procedure will lead to commercially non-available sizes and any attempt to stop gap those values by the closest available commercial sizes can make the design unfeasible and unnecessarily heavier. The desirable stress of an efficient steel truss are based on the minimizing the weight of the structure and keeping the stress on each member under the limits of maximum stress and respective Euler buckling constraint. The optimum design of steel structure subjected to external load is topology dependent and is considered as constrained optimization problem. Weight minimization of structures exposed to stress and displacement constraint is a typical design issue. The design variables can be discrete or potentially continuous and the consideration of the previous one makes the issue harder. Practically, it is desirable to choose design variables which are commercially easily accessible. However, the utilization of a continuous optimization procedure will result in commercially non-accessible sizes and any endeavor to roundoff those values by the nearest accessible commercial sizes could make the design unfeasible or uneconomical.

Genetic algorithm is based on fundamentals of natural evolution of Darwin's theorem. It is an optimization technique most popular due to its simple mathematics and work with fitness and penalties and can handle the continuous and discrete variables. Goldberg and Samtani [1] applied genetic algorithms in optimization of structures. Rajeev and Krishnamoorthy [2] studied GAs in discrete structural optimization of trusses and found out that GAs are suitable for structural optimization since they handle discrete variables efficiently. The improved augmented Lagrangian GA was presented by Adeli [3] as a robust hybrid algorithm for optimization of space structures using the augmented Lagrangian method. Hajela and Lee [4] investigated the application of GA in topological optimization problems and applied its techniques to trusses for stress, buckling, and displacement constraints, showing that the genetic search procedure is a good exploratory tool to evaluate topologies in a discontinuous design space. Additional information on the improvement procedure that produces the structural design from the genotype (a string of bits) and furthermore the fitness assessment procedure of every candidate design is presented in Angeline P. [5]. Chen [6] used GA is an automated design tool with increasing the efficiency by reliability and accuracy of the methodology for code-based design of structures. Deb and Gulati [7] used real-coded genetic algorithms (RCGAs), with specialized reproduction operators, for sizing, topology, and layout optimization of planar and spatial truss structures. Consolidating completely stressed design optimization and conjugate slope methods, shape and cross-segment optimization of trusses is discussed in Gil and Andreu [8]. Krishnamoorthy et al. [9] used an object-oriented framework for GAs in optimization of spatial trusses. Gupta, Ranjan Kumar et al. [10] has shown transformation methods for GA on constrained problems in their work. Azad et al. [11] proposed a mutational based real coded genetic algorithm for sizing and layout optimization of truss with fixed topology and presented classical weight minimization problems of truss structures. Cazauca and Gram [12] proposed a general parameterization and encoding technique to optimize the topology, size and shape of any plane truss by genetic algorithm and finite elements method. They claimed penalty function have great capabilities to express the constraints violations of the solution and leads to faster solution. Weight optimization of plane truss is carried out by Neeraja et al [13] by using GA. Hosseni [14] found the shear capacity of a beam retrofitted by fiber reinforced polymer (FRP) with the help of GA to train the network of artificial neural network. Ede et al [15] presented the analysis of a simple genetic algorithm optimized steel structure, according to BS 5950. Lopes [16] applied genetic algorithm to minimize the concrete volume in foundation. They demonstrated that advanced optimization techniques can be used as auxiliary tools in structural design and concluded that the considerable reduction of volume of concrete is achieved.

In the present study, a Genetic Algorithm which is inspired from the work of Goldberg and Samtani [1] has been used to optimize a simple steel truss. As the problem is of discreet design variables, the GA has been modified and developed using concept of genotypes [5]. Also, as the truss optimized is constrained, therefore a violation of normalized constraints-based formulation for transformation has been used in this paper. The penalty parameter depends on the degree of constraint violation, which is found to be well suited for a parallel search using genetic algorithms [10]. After the development of the suitable GA, it has been analyzed with respect to the different optimization parameters (cross-over methods, rate of mutation) to figure out the fastest method.

2. Genetic algorithm

In a Genetic Algorithm (GA), a predetermined number (a population) of strings(chromosomes) which encode candidate solutions evolves toward a better solution. Genetic algorithms were initially proposed by John Holland at the University of Michigan.

GAs are computationally straightforward and simple, however powerful in their search for optimization. Additionally, they are not constrained by prohibitive assumptions of their search space. GAs are searching systems based over the idea of genetics and natural selection. They utilize the idea of natural selection with genetic operators inspired from nature to build vigorous and powerful search system. GAs vary in various perspectives. Goldberg [5] discusses them in detail.

The different genetic operators that have been distinguished are: reproduction, mutation, speciation, dominance, inversion, deletion, migration, intra-chromosomal duplication, crossover, translocation, segregation, and sharing. Contingent upon the type of the problem and on the prerequisites for performance of the algorithm, GAs can be improved by applying increasingly more of these operators. However, the most straightforward GAs work by utilizing reproduction, crossover and mutation in variable proportions to give the ideal result.

A generic GA can be stated as:

```
begin
Introduce the populace P
Assess each string in the populace
repeat
    repeat
        Select at least 2 individuals in P
        Apply recombination operators with probability pc
        Apply mutation operator with rate pm
        Add new individuals in P'
    until (population P' complete)
    Assess individuals in population P'
    P <- P'
until (termination criteria)
stop
end
```

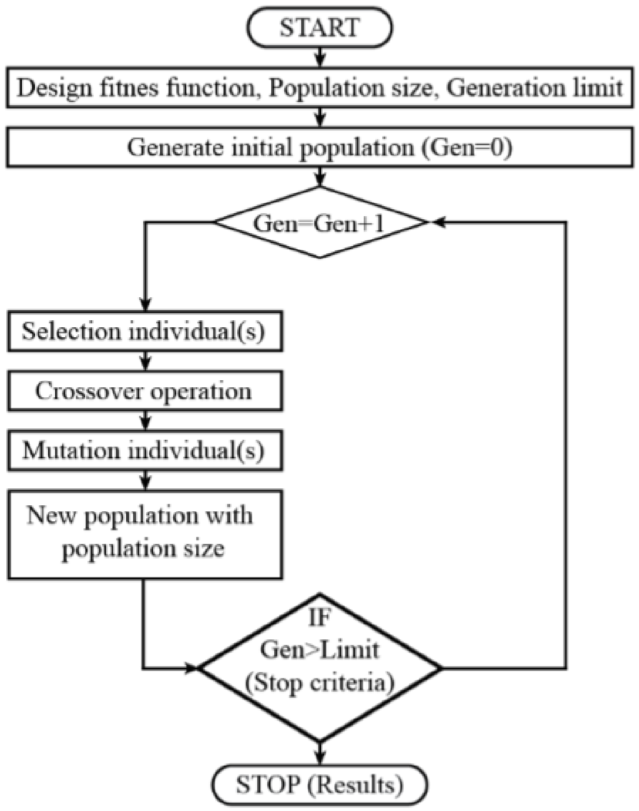
START
Design fitnes function, Population size, Generation limit
Generate initial population (Gen=0)
Gen=Gen+1
Selection individual(s)
Crossover operation
Mutation individual(s)
New population with population size
IF Gen>Limit (Stop criteria)
STOP (Results)

Fig. 1. Flowchart for a typical Genetic Algorithm.

2.1. Procedures in the present work

The methodology of GAs used in this paper is explained below:

2.1.1. The coding procedure

All variables linked to candidate solution encoded in chromosome. In this paper we utilized the binary coding (0 or 1).

2.1.2. The selection scheme

In this paper, a rank-based selection procedure has been adopted. This selection procedure begins by arranging the given populace as indicated by the value of the fitness function and positioning them appropriately. Individual in the populace were then chosen so that high ranking individuals had a higher possibility of being picked for reproduction. We likewise followed Elitism (the top 10% of the populace are constantly duplicated into the next generation directly). This prompted an intermediate populace whose components would then be worked upon.

2.1.3. The recombination operators

The recombination had been cultivated here utilizing three crossover operators – random parent selection, one-point and two-point. The recombination activity is for the most part done with a user defined probability (pc) and with probability (1-pc), activity isn't performed and the two parents are simply replicated and forwarded for mutation step.

2.1.4. The mutation operator

Mutation operator was acquainted to regenerate the errors that might have emerged during duplication procedure. With given mutation rate (low) the mutation operator was applied to each bit in the child chromosomes. If there should be an occurrence of binary digits, the impact of this operator is basic: simply change a 0 into a 1 and the other way around.

2.1.5. The evaluation step

After a new populace has been made, every solution must be assessed so as to have a fitness values assigned to it.

Despite the fact that these operators look basic, their joined activity is liable for a lot of GAs power. From computer programming perspective, they include random-number generation, string-replicating, and halfway string-swapping.

2.2. Genotypes

For discrete design variable, a list of values for the variable is given. There are 16 unique values and 7 design variables (cross sectional area of 7 members). design member connection is done ere by accepting equal cross section area for 1-7;2-6; and 3-5.

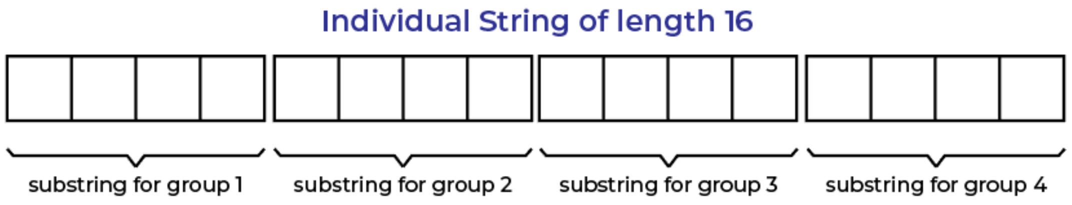

Fig. 2. Individual String (Chromosome).

2.3. Crossover methods

In the following paragraphs we will discuss the Crossover Methods used in this paper.

2.3.1. Single point crossover

Here, a random point of crossover is taken and tails or the heads of the 2 parents are exchanged to obtain new off-springs.

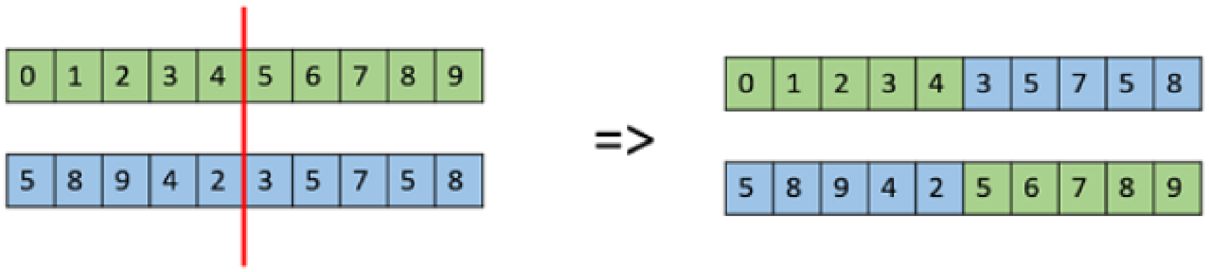

Fig. 3. Pictorial representation of Single Point Cross-over.

2.3.2. Multi point crossover

In Multi point crossover more than one random crossover points are selected and the alternating segments are exchanged to obtain new off-springs.

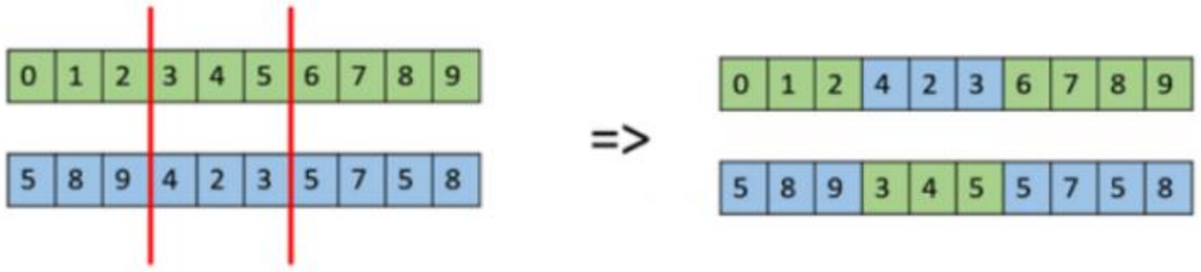

Fig. 4. Pictorial representation of Multi Point Cross-over.

2.3.3. Uniform crossover

Here, we don't separate the parent chromosome into fragments, instead we treat every gene as a fragment. In this, for every gene we randomly decide if it'll be included in the off-spring. We can likewise attribute this randomness to one parent.

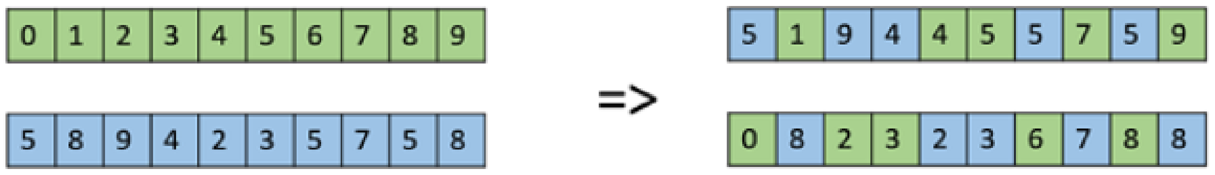

Fig. 5. Pictorial representation of Uniform Cross-over.

2.4. Mutation

It is characterized as a random change in the chromosome, to obtain another solution or derived chromosome. It is utilized to present assorted variety in the genetic populace and is typically used with lesser probability. It is discovered that mutation is fundamental for the GA's convergence.

2.4.1. Bit flip mutation

Here, we select at least 1 bit at random and flip them. This is utilized in binary encoded GAs. This is the mutation method which has been used in this paper.

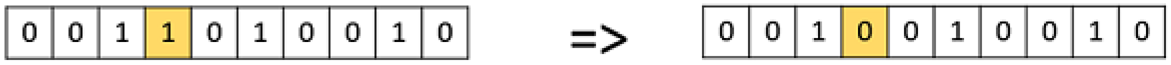

Fig. 6. Pictorial representation of Bit Flip Mutation.

The genetic algorithm rehashes a similar procedure again and again by creating new populace and assessing its fitness. In spite of the fact that there are numerous random activities in GA's, they don't discover the solution randomly.

For each new "child "solution for be created, a pair of "parent" solution is chosen. A "child" solution is created utilizing genetic operators, which normally shares huge numbers of the attributes of its "parents". Now, new parents are chosen from each new child, and the procedure proceeds until another populace of arrangements of appropriate size is produced. The normal fitness of the new generations is relied upon to be higher than that of the past generations, as the best individual of the last generations have been given higher possibilities for breeding. The algorithm will end when either the most extreme number of generations given has been produced, or a satisfactory fitness level has been accomplished for the populace.

3. Optimization of steel truss

3.1. Example problem

We have considered a simple three-bar truss as shown in Fig. 1 to explain the optimization procedure. The data for the truss are: Maximum stress (σM) = ±147.15 MPa, density d = 7.85 x 103 kg/m^3; modulus of elasticity E = 2.008 x 10^5 MPa; and Euler buckling constraint (σE) = ±4EA/L^2. We need to get optimum c/s area of the truss members 1, 2, 3 and 4(fig.1) for the given loading conditions and assuming the self-weight of each member is distributed equally and applied on both ends of the member. Mathematically: minimize f(x) for g_j{x) < 0, j = 1,2,3, . . . where, j is the number of constraints.

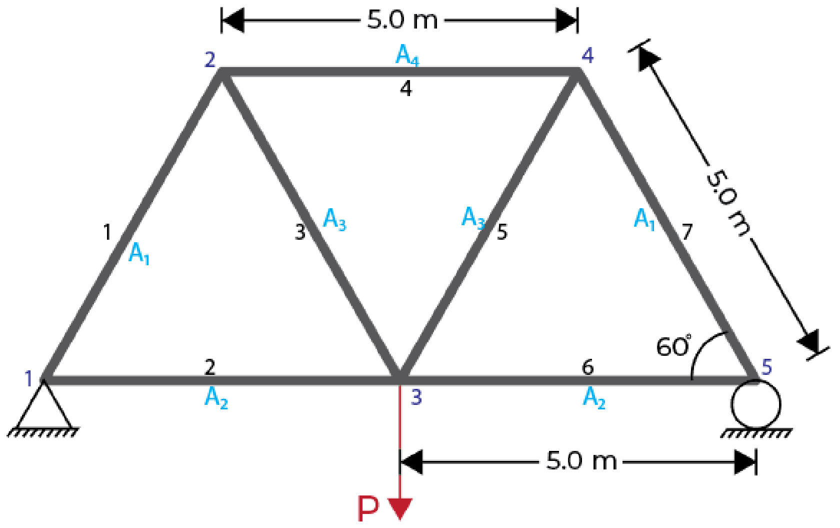

Fig. 7. Steel Truss to be optimized.

3.1.1. Mathematical formulation

$f(x)=\sum dA_jL_j$

where A_j = the c/s area of jth member; L_j = length of jth member;

d = the weight density of material;

$\sigma_j<\sigma M$, for j = 1, 2, 3

$\sigma_j<\sigma E_j$, for j = 1, 2, 3

where σ_j = stress in member j; σM = allowable stress and σE_j= Euler Buckling constraint of member j.

In this problem, we cannot describe all the constraints in terms of the design variables; because they are implicit and their evaluation would require analysis of the truss structure. We know that, Genetic algorithms works better for unconstrained problems and as the presented problem here is a constrained one, it is required to transform it into an unconstrained one, so it could be optimized using GAs.

$(\sigma_j / \sigma M)- 1< 0$; and $(\sigma_j /\sigma E_j) -1 < 0$

A violation coefficient C is given as: if $g_i(x) > 0$, then $c_i = g_j(x)$; or if $g_i(x) <= 0$, then $c_i = 0$,

and $C = \sum c_j$

Now the modified objective function $\phi(x)$ is written as

$\phi(x) =f(x) (1 + KC)$

K=10 in this paper.

Now ϕ(x) will be the fitness function (or objective function) for our Genetic Algorithm which will be used to carry out the unconstrained optimization. In our case we are going to terminate the algorithm when the fitness values of the fitness function reach a satisfactory level and becomes consistent. We check for the concurrence of the fitness values for at least 10 consecutive generations as a mark for terminating the algorithm.

Our fitness function also becomes equal to the weight of the truss when none of the members are under stress that is beyond the limiting stress or the Euler Buckling constraint. Thus, for the final result to be stable and feasible, the value of the fitness function must be equal to the value of the weight of the truss structure.

3.2. Optimization of the problem using different variations of genetic algorithm in the present work

1. The first program follows a uniform crossover GA in which every new offspring is made by randomly selecting the parent from which the gene is taken. The mutation rate was kept at 2% and 10% elitism was followed.

2. The second program follows two-point (multi) crossover between the fifth and eleventh indices. The mutation rate was kept at 2% and 10% elitism was followed.

3. The third program was written using single point crossover after the seventh index of the chromosome and taking a mutation rate of 2% and elitism was kept at 10%.

4. The fourth program follows multi-point crossover between the fifth and eleventh indices. The mutation rate was varied to see its effect on the number of generations required for optimal solution.

4. Results

4.1. Using uniform crossover

Parameters used:

Stress limit = 147.150 MPa
Modulus of Elasticity = 2.008 x 105 MPa
Density = 7.85 kg/m3
Length of the base member = 5.0 m
Constraint Violation penalty constant = 10
Force on Joint 3 = 100 N
Mutation rate = 2%
Elitism = 10%

The optimal solution came out to be:

Table 1
Optimal Solution using uniform crossover.

A_1 (cm^2)	A_2 (cm^2)	A_3 (cm^2)	A_4 (cm^2)	Weight(kg)
3.4	2.4	2.4	2.8	753.6

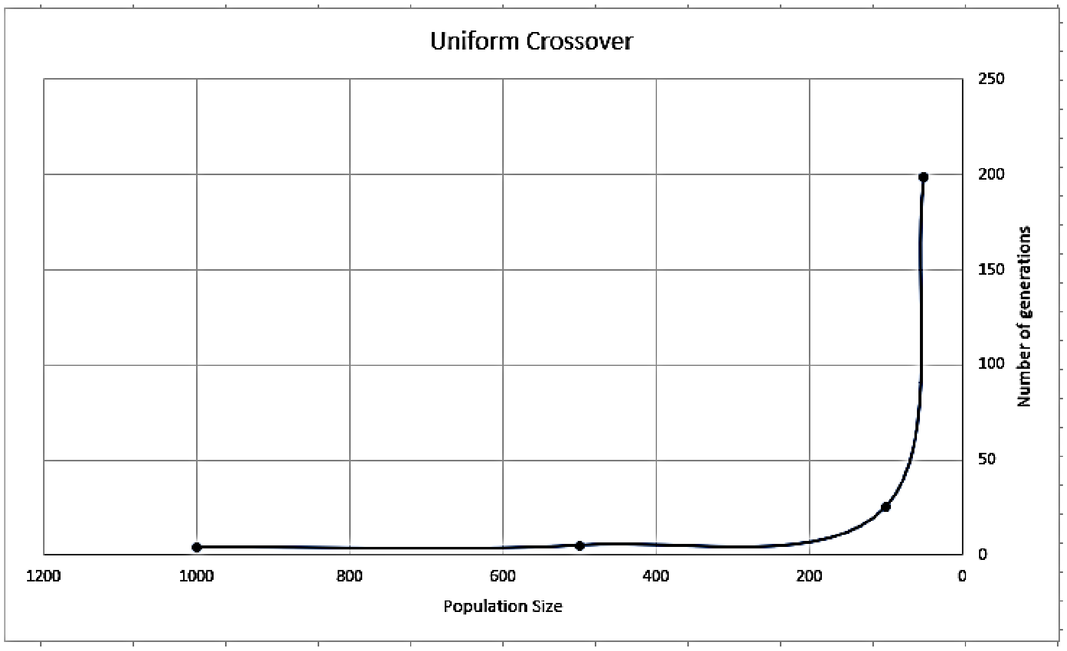

Fig. 8. Variation in number of generations to achieve optimal solution with population size using uniform crossover.

As we can see, when we used the Uniform crossover for various population sizes, the number of generations taken to get the optimal solution for different initial population sizes varies exponentially. As we go on decreasing the population size the number of generations increases first gradually then with a very quick pace.

4.2. Using multi point crossover

Parameters used:

Stress limit = 147.150 MPa
Modulus of Elasticity = 2.008 x 105 MPa
Density = 7.85 kg/m^3
Length of the base member = 5 m
Constraint Violation penalty constant 10
Force on Joint 3 = 100 N
Mutation rate = 2%
Elitism = 10%

The optimal solution came out to be:

Table 2
Optimal Solution using Multi Point crossover.

A_1 (cm^2)	A_2 (cm^2)	A_3 (cm^2)	A_4 (cm^2)	Weight(kg)
3.4	2.4	2.4	2.8	753.6

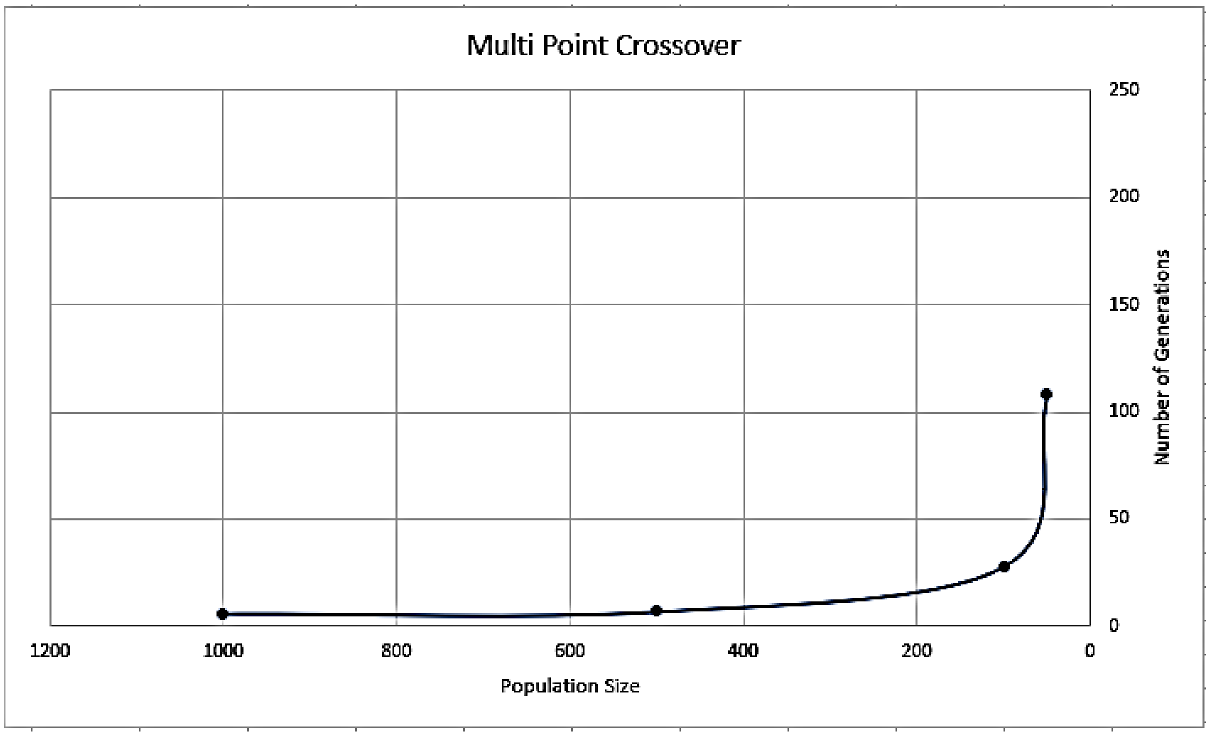

Fig. 9. Variation in number of generations to achieve optimal solution with population size using multi point crossover.

In this we used multi-point crossover for the mating function by taking the elements – 1 to 5 and 12 to 16 from the first parent and the elements 6 to 11 from the second parent to make the child chromosome. As we can see, when we used the Uniform crossover for various population sizes, the number of generations taken to achieve the optimal solution for the problem for different initial population sizes varies exponentially. As we go on decreasing the population size the number of generations increases first gradually then with a very quick pace.

4.3. Using single point crossover

Parameters used:

Stress limit = 147.150 MPa
Modulus of Elasticity = 2.008 x 105 MPa
Density = 7.85 kg/m3
Length of the base member = 5 m
Constraint Violation penalty constant = 10
Force on Joint 3 = 100 N
Mutation rate = 2%
Elitism = 10%

The optimal solution came out to be:

Table 3
Optimal Solution using Single Point crossover.

A_1 (cm^2)	A_2 (cm^2)	A_3 (cm^2)	A_4 (cm^2)	Weight(kg)
3.4	2.4	2.4	2.8	753.6

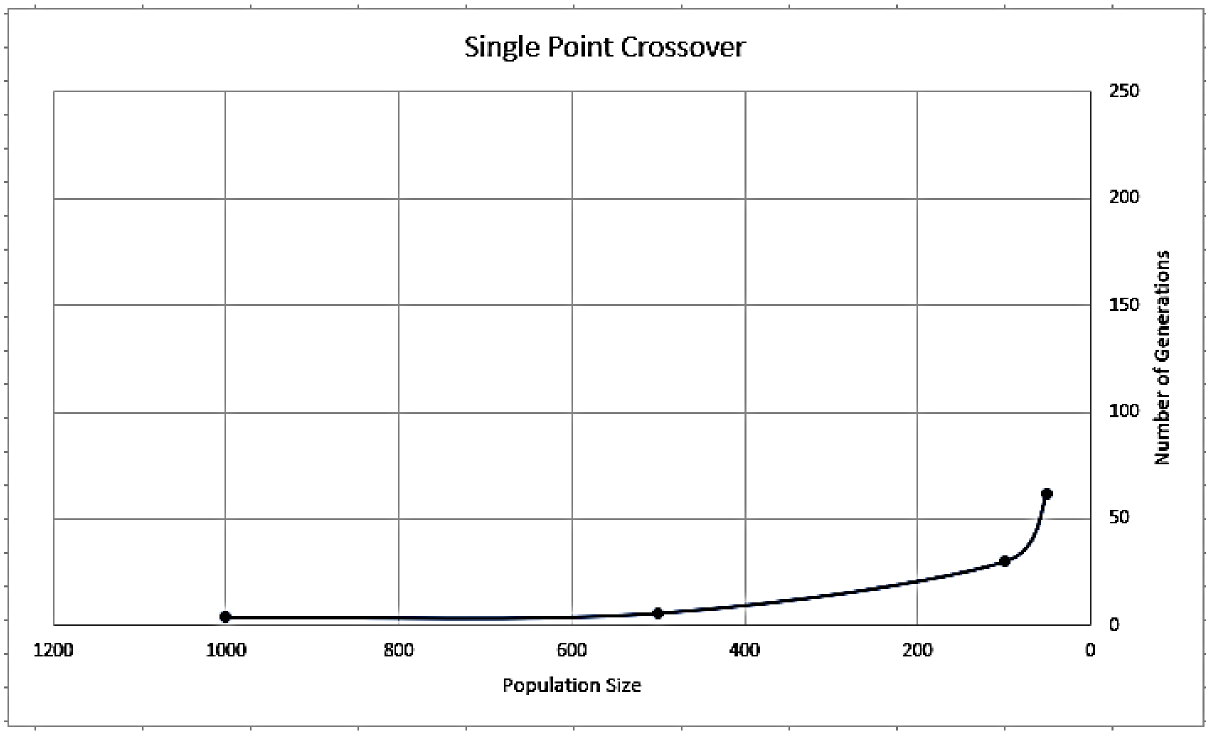

Fig. 10. Variation in number of generations to achieve optimal solution with population size using single point crossover.

In this we used single point crossover for the mating function by taking the elements – 1 to 8 from the first parent and 8 to 16 from the second parent to make the child chromosome. As we can see, when we used the single point crossover for various population sizes, the number of generations taken to achieve the optimal solution for different initial population sizes varies exponentially. As we go on decreasing the population size the number of generations increases first gradually then with a very quick pace.

4.4. Effects of increasing the rate of mutation

In the above cases, we changed the crossover methods, keeping the mutation rate constant at 2%. In this section we'll observe the change in the number of generations with increase in the rate of mutation, keeping the crossover method as Multi point crossover and population size constant at 50.

Table 4
Variation of number of generations to achieve optimal solution with rate of mutation.

Rate of Mutation	2	5	10	20
Number of Generations	117	92	92	147

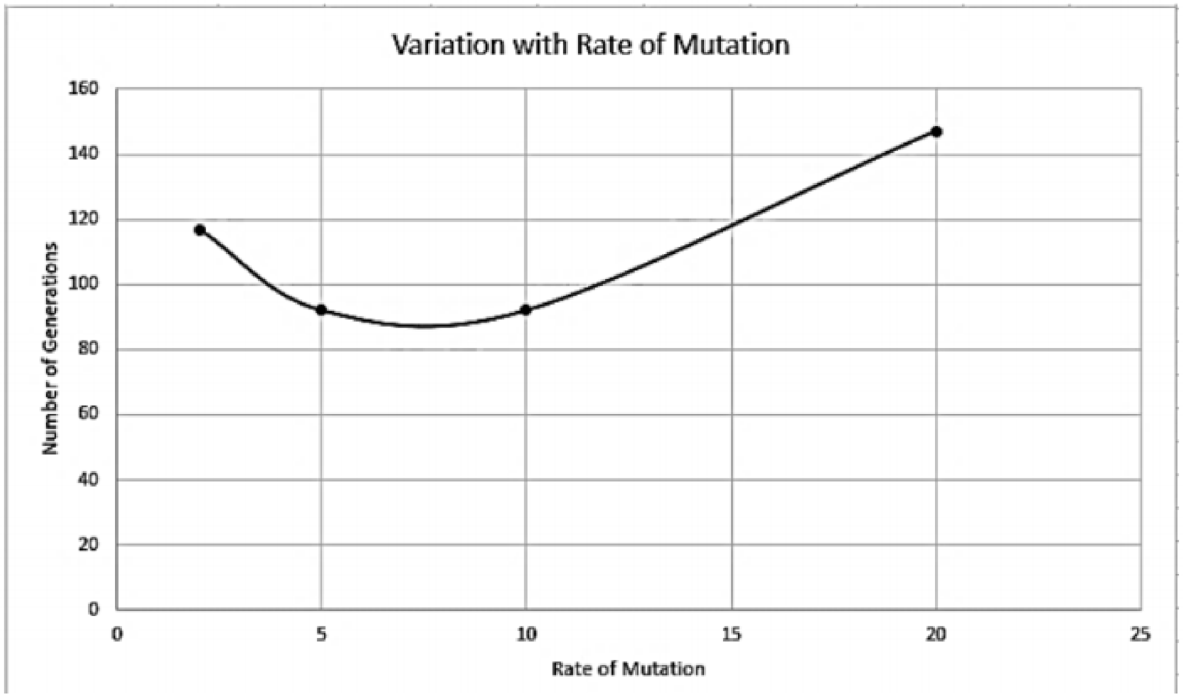

Fig. 11. Variation in number of generations to achieve optimal solution with rate of mutation.

As we can see, the number of generations first decrease with increase in the mutation rate and as we go on increasing the rate of mutation, the number of generations start increasing as the randomness of the chromosomes of the offspring increases and this affects the algorithm in more negative ways than positive. Thus, slowing down the optimization process.

5. Conclusions

In this paper various GAs are used to solve the weight minimization problem for a simple truss consisted of steel members. . It is quite evident that GA is one of the best choices available for discrete optimization of structures or any other such cases because of its ease of application and performance. The application of genetic algorithm using binary encoding for our problem was very simple.

As compared to continuous optimization, discrete optimization gave us results that are a lot more practical and ready to use as we choose amongst the options that are available in the market. The penalty-based transformation method (i.e., violation of normalized constraints-based formulation) which was used to transform our constraint problem into an unconstraint one (which resulted in the modified objective function for our problem) worked very well, which can be implied from the obtained results. The secondary aim was to explore the sensitivity of the Genetic Algorithm to the operator values used. Different variations of Genetic Algorithm (i.e., different crossovers, different rate of mutation) were tried as a separate case and the effect of that can be seen in the results. All of them, however, resulted in same optimal values for the design of the structure i.e., cross-section of members and weight of the truss. For a large population size the differences in the speed of these methods are negligible, but at lower population size we can easily observe that the number of generations taken for obtaining optimized results was lowest in Single Point Cross-over followed by Multi Point and Uniform cross-over method which implies that Single Point Cross-over was fastest.

References

[1] Goldberg DE, Samtani MP. Engineering optimization via genetic algorithm. Electron. Comput., ASCE; 1986, p. 471–82.

[2] Rajeev S, Krishnamoorthy CS. Discrete Optimization of Structures Using Genetic Algorithms. J Struct Eng 1992;118:1233–50. doi:10.1061/(ASCE)0733-9445(1992)118:5(1233).

[3] Adeli H, Cheng N. Concurrent Genetic Algorithms for Optimization of Large Structures. J Aerosp Eng 1994;7:276–96. doi:10.1061/(ASCE)0893-1321(1994)7:3(276).

[4] Hajela P, Lee E. Genetic algorithms in truss topological optimization. Int J Solids Struct 1995;32:3341–57. doi:10.1016/0020-7683(94)00306-H.

[5] Angeline PJ. Morphogenic Evolutionary Computations: Introduction, Issues and Example. Evol. Program., 1995, p. 387–401.

[6] Chen S-Y, Rajan SD. A robust genetic algorithm for structural optimization. Struct Eng Mech 2000;10:313–36.

[7] Deb K, Gulati S. Design of truss-structures for minimum weight using genetic algorithms. Finite Elem Anal Des 2001;37:447–65. doi:10.1016/S0168-874X(00)00057-3.

[8] Gil L, Andreu A. Shape and cross-section optimisation of a truss structure. Comput Struct 2001;79:681–9. doi:10.1016/S0045-7949(00)00182-6.

[9] Krishnamoorthy CS, Prasanna Venkatesh P, Sudarshan R. Object-Oriented Framework for Genetic Algorithms with Application to Space Truss Optimization. J Comput Civ Eng 2002;16:66–75. doi:10.1061/(ASCE)0887-3801(2002)16:1(66).

[10] Gupta RK, Bhunia AK, Roy D. A GA based penalty function technique for solving constrained redundancy allocation problem of series system with interval valued reliability of components. J Comput Appl Math 2009;232:275–84. doi:10.1016/j.cam.2009.06.008.

[11] Kazemzadeh Azad S, Jayant Kulkarni A. Structural optimization using a mutation-based genetic algorithm. Iran Univ Sci Technol 2012;2:81–101.

[12] Cazacu R, Grama L. Steel Truss Optimization Using Genetic Algorithms and FEA. Procedia Technol 2014;12:339–46. doi:10.1016/j.protcy.2013.12.496.

[13] Neeraja D, Kamireddy T, Kumar PS, Reddy VS. Weight optimization of plane truss using genetic algorithm. IOP Conf Ser Mater Sci Eng 2017;263:032015. doi:10.1088/1757-899X/263/3/032015.

[14] Hosseini G. Capacity Prediction of RC Beams Strengthened with FRP by Artificial Neural Networks Based on Genetic Algorithm. J Soft Comput Civ Eng 2017;1:93–8.

[15] Ede AN, Oshokoya OO, Oluwafemi JO, Oyebisi SO, Olofinnade OM. Structural analysis of a genetic algorithm optimized steel truss structure according to BS 5950. Int J Civ Eng Technol 2018;9:358–64.

[16] Lopes M, Pires Soeiro FJ, Santos Da Silva JG. Structural Optimization of Concrete Volume for Machine Foundation Using Genetic Algorithms. J Soft Comput Civ Eng 2019;3:62–81.